2023-01-0040

Cybersecurity Vulnerabilities for Off-Board Commercial Vehicle Diagnostics

Author, co-author (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Affiliation (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Abstract

The lack of inherent security controls makes traditional Controller Area Network (CAN) buses vulnerable to Machine-In-The-Middle (MitM) cybersecurity attacks. Conventional vehicular MitM attacks involve tampering with the hardware to directly manipulate CAN bus traffic. We show, however, that MitM attacks can be realized without direct tampering of any CAN hardware. Our demonstration leverages how diagnostic applications based on RP1210 are vulnerable to Machine-In-The-Middle attacks. Test results show SAE J1939 communications, including single frame and multi-framed broadcast and on-request messages, are susceptible to data manipulation attacks where a shim DLL is used as a Machine-In-The-Middle. The demonstration shows these attacks can manipulate data that may mislead vehicle operators into taking the wrong actions. A solution is proposed to mitigate these attacks by utilizing machine authentication codes or authenticated encryption with pre-shared keys between the communicating parties. Various tradeoffs, such as communication overhead encryption time and J1939 protocol compliance, are presented while implementing the mitigation strategy. One of our key findings is that the data flowing through RP1210-based diagnostic systems are vulnerable to MitM attacks launched from the host diagnostics computer. Security models should include controls to detect and mitigate these data flows. An example of a cryptographic security control to mitigate the risk of an MitM attack was implemented and demonstrated by using the SAE J1939 DM18 message. This approach, however, utilizes over twice the bandwidth as normal communications. Sensitive data should utilize such a security control.

Introduction

Medium and Heavy Duty (MHD) network communication mostly takes place over SAE J1939 networks, which are built on Controller Area Network (CAN) 2.0b compliant networks with extended arbitration identifiers. These are multi-master serial bus networks where nodes are physically connected by two twisted pair wires and a high-speed CAN transceiver. SAE J1939 is the recommended vehicle bus practice used for diagnostics and communication among vehicle components on MHD vehicles. Originating in heavy duty truck industry in United States, J1939 is now widely used in other parts of the world. J1939 is found wherever a diesel engine may be used for power. SAE J1939 features both unicast (destination-specific) and broadcast messages and as well supports transport fragmentation/reassembly and address claiming. However, most of the critical operations such as reconfiguration and programming are protected by a seed-key exchange, which is a challenge-response system between the electronic control unit (ECU) and the diagnostics software application.

In MHD vehicles there are often many different ECU suppliers and each of them may have a different diagnostic application to interface with the ECU over the in-vehicle network. The diagnostic application is often running on a Windows-based PC or laptop and the ECU is on the in-vehicle network, like J1939. This means there must exist a device to translate the communications on the vehicle network to the PC/laptop. In the context of MHD vehicles, these are known as vehicle diagnostic adapters (VDAs).

Early in the electronification of MHDs, each component manufacturer would specify a unique or proprietary VDA for their diagnostics. Fleet owners would have to maintain many of these different adapters based on the configuration of their vehicles. Since managing and acquiring multiple VDAs was a pain-point for owners and maintainers, the American Trucking Association's (ATA) Technology and Maintenance Council (TMC) initiated Recommended Practice (RP) number 1210 in the 1990's. The RP1210 is titled Windows Communication API, where API stands for application program interface. The purpose of RP1210 is to describe a standard API for Windows PC applications to communicate with the in-vehicle network.

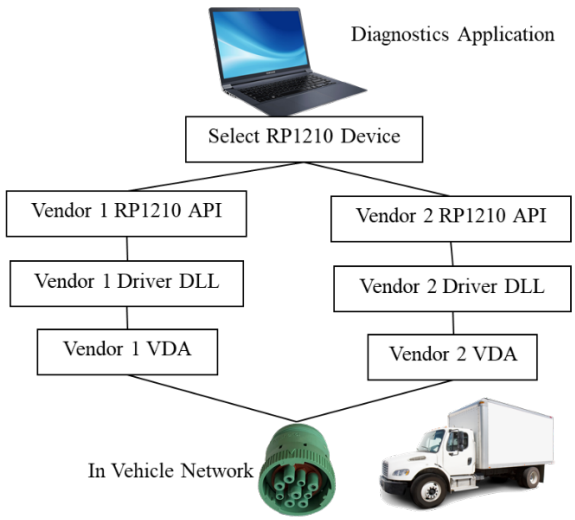

Figure 1: The concept of RP1210 where the diagnostic application can select between different vendors of the vehicle diagnostic adapter (VDA).

The RP1210 concept is explained in Figure 1 by showing the ability for the user of the diagnostic application to select an RP1210 compliant device. The choices for the user to select are kept in an RP1210.ini file, which is specified in the recommended practice. These selections point to the individual vendor .ini file to give the user the specified valid options for each device. Once a vendor and device are selected, there is a specific device API that accepts and presents the interface to the diagnostic application. The device driver connects the RP1210 function prototypes to the vendor specific mechanisms for communication with their vehicle diagnostic adapter through the vendor DLL.

It is important to note that the VDA vendors control the firmware on the VDA device, the update mechanism for the VDA firmware, the RP1210 drivers, and device drivers on the PC. These conduits for diagnostic and maintenance communications in MHDs are inherently trusted by the diagnostics software connected to the electronic control unit. Therefore, this paper explores the cybersecurity of the software and communications stack within the diagnostics PC/laptop computer.

SAE J1939 Protocol

In heavy-duty vehicles, the SAE J1939 protocol establishes standards defining how connected nodes should communicate on a CAN bus. J1939 is a 29-bit identifier with a 3-bit priority that decides the arbitration the message achieves on the bus, 0 being the highest priority. The next 1-bit is reserved for future use, and the 1-bit post this bit is a data page that can be used to expand PGN. Followed by the data page is an 8-bit PDU format the indicates whether the message is broadcast or point-to-point. The 8-bit PDU-specific field has two meanings corresponding to the PDU format. If the message is a broadcast, this field contains the group extension of PDU format if between 0 and 239. This field shall have destination address if this is a point-to-point message. The 29-bit identifier ends with 8-bit address of the source that transmits the message. On classic CAN, the payload is 8 bytes where SAE J1939 protocol is tunneled.

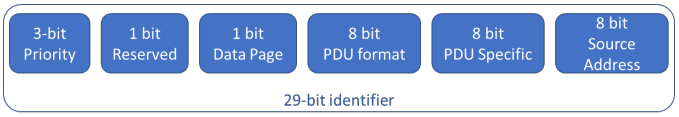

Figure 2: SAE J1939 Structure of a CAN ID

Vehicle Diagnostics

Cybersecurity assessments for vehicles have often been scoped around the driving operation of the vehicle and the system boundary shows the vehicle on its wheels. This picture is incomplete, because during the lifecycle of the vehicle system, there will likely be external connections using service and diagnostic tooling. These are intermittent connections, yet they are highly trusted. A trusted maintenance technician is often granted access to connect a vehicle diagnostic adapter (VDA) to the diagnostic port and exercise the off-board communications to read and write data. The writing of data can be as simple as requesting information to perform a complete reflash of an on-board ECU. Compared to passenger vehicles, commercial vehicles are more frequently connected to this diagnostic equipment making the threat larger for commercial vehicles.

Many diagnostic and maintenance operations that update calibrations of ECUs are protected by a seed-key exchange. These exchanges may use a 2-byte seed and look for a corresponding 2-byte key to authenticate the session. However, this only happens at the beginning of the session and none of the diagnostic traffic flowing through the RP1210 stack and through the in-vehicle network is further authenticated. This means it is trivial for a process to co-opt or highjack the diagnostic session, even after a legitimate seed-key exchange.

Because the RP1210 stack is situated in the middle of the communications between the diagnostic application and the ECU client application, the ability to perform deep packet inspection and manipulation of all traffic, both to and from the ECU, is possible if those communication paths are compromised. This paper demonstrates how to compromise the RP1210 communications and presents a viable method to mitigate the effects of a compromise.

Vehicle Network Attacks

This section focuses on some of the attacks in the literature that demonstrate some cybersecurity attacks described in the literature. This is not an exhaustive review but should suffice to convince the reader that cyberattacks can be launched against MHD vehicles. An important feature of these attacks is they assumed connectivity to the network. In the context of a diagnostics session, a trusted technician is creating the network connectivity for the attacker who would work through the RP1210 stack.

Among the various communication protocols found in automotive networks such as LIN, FlexRay, CAN-FD, Automotive Ethernet etc. the most widely used is a Controller Area Network (CAN) based multi-master serial bus as it is a shared bus and hence cheaper. The CAN bus vehicle bus standard is a message-based protocol, originally designed to save on copper by multiplexing electrical wiring within automobiles. Hence, while designing the CAN protocol, security was considered less. Due to the inherent vulnerabilities of CAN, Ref. [2] demonstrates how simple it is to disrupt the safety critical features of running vehicle using simple message injection techniques. These messages can be well formed and the J1939 network cannot tell the difference between a legitimate message and one that has been injected by an attacker.

On a CAN bus, data in transit is not secure as the communication is unencrypted or authenticated. Even if the attacker lacks the ability to craft a message, vehicle operations may be vulnerable to replay attacks that impact vehicle operation. An example is an operator losing the ability to close a window when at a speed above 200 km/h [3]. Further, the safety and performance of vehicles platooning is impacted when replay attacks are executed on unencrypted corporative adaptive cruise control messages [4]. By hijacking a unified diagnostic service (UDS) session using a tampered vehicle announcement message by poisoning the address map, an attacker gains all privileges of a legitimate UDS session [5]. Attackers may be able to directly interact with safety-critical features such as autopilot if the protocol is not authenticated and if the session is unencrypted [6]. This not only opens doors for identity spoofing, unauthorized access, and MitM attacks but also facilitates the leaking of sensitive information. Ref. [7] discusses how safety and security are intertwined, which makes diagnostics security a safety issue.

The widely used SAE J1939 recommended practice defines the transport layer in SAE J1939-21 to transmit and receive data payloads that are greater than 8 bytes (and less than 1785 bytes). To avoid resource exhaustion, there is usually an upper limit on the

maximum number of such connections. However, as the transport connections are unauthenticated, an attacker can establish the maximum number of parallel connections and make them stay alive by transmitting a falsified Request to Send (RTS) message and hence creating a connection exhaustion [8]. As the CAN protocol lacks sender authentication, by inserting a physical malicious gateway device in between two ECU's communication line an active attacker can add, modify or even delete messages going back and forth [9]. A vehicle diagnostics session attack using VDA firmware, PC driver and a middle-person attack were demonstrated, along with secured diagnostics gateway mitigation [10]. With a minimal modification of a diagnostic application, a Stuxnet-style MitM stealthy attack to switch off the passenger airbag was demonstrated [27].

This paper contributes to the body of knowledge related to vehicle cybersecurity by archiving the technique and mitigation for conducting a cyberattack by compromising the RP1210 communication used by heavy vehicle diagnostics systems. A similar approach is feasible by using an SAE J2534-compliant service tool for passenger cars.

Proposed Defenses

Mitigating a MitM attack is well-researched and known in Information Technology (IT) networks using technologies like virtual private networks (VPNs) and transport layer security (TLS), these approaches are out of scope for this paper, since the focus is on vehicle diagnostics for MHD vehicles using SAE J1939. This section describes some of the approaches in the literature to mitigate the effect of compromising a vehicle connection with a machine-in-the-middle.

Using application layer encryption of the SAE J1939 diagnostic traffic between the vehicle diagnostic application and the in-vehicle secure gateway diagnostic attacks were mitigated in [12]. To reduce the risk of leakage of pre-shared keys and to support forward secrecy, keys are dynamically generated using elliptic curve Diffie-Hellman (ECDH) key exchange, which was used for encryption. Existing shortcomings in the SAE J1939 specifications gave rise to several new attacks, e.g., impersonation, denial of service (DoS), distributed DoS, etc. with potential safety critical effects while still conforming to the SAE J1939 standard specification. Ref. [13] recommends mitigation mechanisms by including message authentication. By combining the timing analysis with a packet manipulation detection system, Ref. [14] captures the state of the vehicle, detects messages with irregular timing intervals, and takes advantage of the dependencies between different ECUs to restrict attackers.

While cyber-attacks can be detected upon detection rules or by statistical analysis of timing regularities, [15] proposes a Feistel Cipher Block based MitM defense by encrypting diagnostic communication with pre-shared key. By using a bit-banged Controller Area Network (CAN) filter, attacks can be detected before the message finishes transmitting and calculating the cyclic redundancy check specified in the CAN protocol [16]. Once an attack is discovered, the defender induces a CAN protocol error to invalidate the malicious message from the network and it never reaches the intended application.

Often existing or older electronic control modules have constrained processors in processing power, memory size, and especially with respect to cybersecurity features. An inadequate entropy source used for cryptographic purposes opens doors for reverse engineering. For instance, a poor TRNG used in a seed/key authentication reduces the brute force attack space and hence the brute force time to compromise a session authentication. Ref. [17] articulates how to overcome the lack of a hardware-based true random number generator by creating an entropy source by combining several sources of entropy such as analog to digital (AD) channels, ECU configurations and other sources of randomness. However, the entropy property of the seed/key exchange is irrelevant to the MitM attack because the attacker can let the legitimate service tool perform the authentication then hijack the communications.

Paper Organization

The rest of the paper is organized as follows. The attack demonstration section discusses the attack model, experimental setup used and incisive categorization of attacks to prove that the attack is possible on all J1939-based communication. Mitigation demonstration walks through the mitigation developed also indicating effectiveness of following the principles of security by design as recommended in ISO/SAE 21434 [18]. The final section concludes the paper with an overview of the results achieved and future direction of the research in terms of attacks and defenses.

Attacking Vehicle Diagnostic Adapter Drivers

The Technology and Maintenance Council (TMC) recommended practice RP120 is used for analyzing and reprogramming Electronic Control Units (ECUs) in heavy duty vehicles. Using Microsoft Windows operating system, RP1210 defines standard APIs for communication between ECUs and a PC thus enabling interoperability with various hardware interfaces. An abbreviated overview of the standard RP1210 function prototypes are tabulated in Table 1.

Table 1. Function prototypes exposed froma n RP1210 compliant API.

Function Name	Description
`LoadLibrary`	Open the VDA API's library.
`RP1210_ClientConnect`	Connect to the vehicle data bus.
`RP1210_SendCommand`	Send commands to the VDA
`RP1210_SendMessage`	Send a message on the network.
`RP1210_ReadMessage`	Read a message from the network.

Vehicle diagnostic adapter hardware often supports network protocols such as CAN, J1587, J1708, J1939, J1850, ISO15765 to communicate to the ECU. These protocols each have their own specification within RP1210. In other words, a CAN message has a different structure than a J1939 message in RP1210. The data communication between an RP1210 device and PC can be via COM serial port, LPT parallel port, PCMCIA card device, USB device, TCP/IP, Bluetooth, WiFi, or any other low-level communication mechanism between the embedded device on the VDA and the PC/laptop.

Unlike the Linux operating system, Windows has no clearly defined interfaces for applications to interact with the Kernel. Instead, Windows provides several user space dynamic link libraries (DLLs) for applications to interact with the Kernel. When an RP1210 diagnostic adapter driver is installed, an RP1210 API DLL file with a unique name is created. An authorized diagnostic software application can use this unique RP1210 API DLL name to select the RP1210 hardware to use. Thus, this specific vendor RP1210 API

DLL provides a link between protocol specific API and RP1210 API functions as shown in Figure 3.

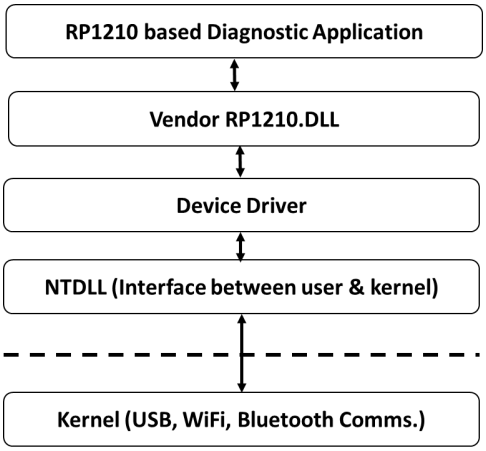

Figure 3: Communication stack within the PC/laptop.

The RP1210 document is publicly available and contains enough information for a programmer to utilize and create function prototypes available to a normal diagnostics application. Therefore, a so-called shim windows application can be developed to interact with an RP1210 based diagnostic application via standardized APIs. After listing the Windows installation folders, the unique name of the authorized RP1210 API DLL is understood. An attacker can rename a malicious shim windows application DLL developed to the unique name of the authorized RP1210 API DLL, while the existing authorized is renamed with a unique name that will be used in the malicious windows application. For example, there is a VDA drive for Vendor A installed with a DLL named VENDOR_A_32.DLL. This file is renamed to ORIG_VENDOR_A_32.DLL and a new program (DLL) is installed with the original name of VENDOR_A_32.DLL. Since the RP1210 ini file is not updated, the user has no indication that the newly created file is potentially malicious.

By performing this file renaming, an attacker successfully redirected the authorized communication channel and inserted a malicious shim windows application in the middle as shown in Figure 4. The malicious application is now capable of transparently intercepting, manipulating, and resizing an authorized RP1210 message. If the RP1210 network client is CAN, then the ID, length and data can be changed. Similarly, if the RP1210 network client is J1939, then the priority, parameter group number (PGN), destination address DA), source address (SA), length, and data can all be manipulated. The stealthy malicious shim application now has all the privileges of an authorized diagnostic application.

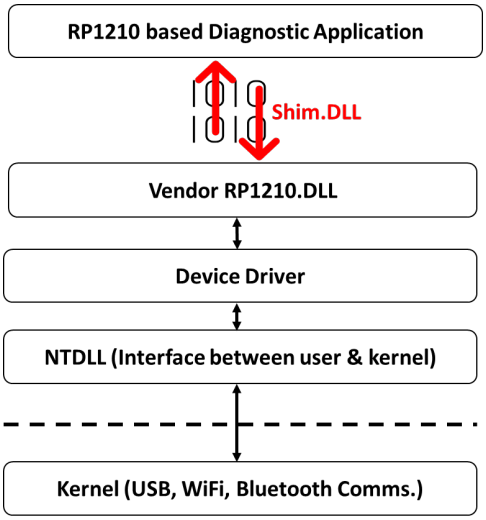

Figure 4: Inserted shim DLL able to affect RP1210 communications.

To establish an authorized diagnostic session, the RP1210 based diagnostic application may process a secret key. Even though the secret key is unknown to the malicious shim application, it can closely monitor the communication channel and hijack the session when session authentication is completed. The opportunities of tampering using such an inserted shim are immense, but this paper is focused on demonstrating that tampering is possible on J1939 communication that undermines security.

Security Experiment Setup

A shim DLL was written in Visual Studio 2022 as a console application. The partial source code is available in the appendix. The design of the shim is to open an RP1210 connection with a legitimate vendor DLL and expose the needed function prototype for the diagnostic software. The first instance of this shim dll is to simply pass the data faithfully from one function call to another. For example, the shim would implement and expose RP1210_ReadMessage() to interface with the legitimate vendor's DLL function for RP1210_SendMessage(). Similarly, the shim DLL would implement and expose RP1210_SendMessage() to interface with the legitimate vendor's DLL function for RP1210_READMessage(). In this manner, the shim DLL is a simple passthrough application. Logging and exfiltrating functions are trivial to implement in such a shim. Covertly building a library of logged data would be useful for reverse engineering a component's diagnostic protocol or vehicle utilization.

All attacks were conducted on a bench-setup consisting of a 500-kbps baud rated CAN bus connecting an ECU and RP1210 adapter via a Deutsch 9-pin connector as specified in J1939-13. The VDA was connected using USB for the setup. RP1210 drivers and an RP1210 diagnostic application from manufacturer X were installed on the diagnostics PC. Normal connectivity to the ECU was verified on the diagnostic tool before inserting the shim DLL. After inserting the passthrough shim DLL in passthrough mode, normal connectivity was still observed. A photograph of the test setup is shown in Figure 5.

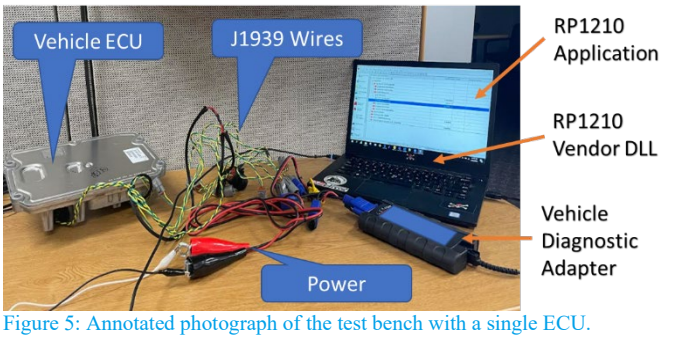

Figure 5: Annotated photograph of the test bench with a single ECU.

Attack demonstration on SAE J1939 messages

Many messages in J1939 are 8 bytes long, which is the limit of CAN 2.0b. However, the transport protocol defined in SAE J1939-21 enables J1939 to handle messages with a length of 9 to1785 bytes using multiple frames. Construction and deconstruction of long messages in J1939 can be implemented in the RP1210 device driver or in VDA firmware and is typically done in the RP1210 device driver. J1939 and RP1210 support destination-specific transfers using connection management messages. Destination-specific transfers achieve handshaking using Request to Send (RTS) and Clear to Send (CTS) messages. An RP1210 Vehicle Datalink Adapter (VDA) should de-packetize/packetize and not send the individual Transport Protocol (TP) packets to the application if nIsAppPacketizing in RP1210_ClientConnect() is set to FALSE. However, if this is set to TRUE the VDA shall not de-packetize/packetize the TP packets. For this demonstration we had it set to FALSE. The requirements and services required to enable an ECU on a network segment to intercommunicate with devices on a different network segment are described in the SAE J1939-31work layer. The PGN data field parameter placement notations and conventions known as Suspect Parameter Number (SPNs) are specified in SAE J1939-71 Vehicle Application Layer. The details of messages (PGNs), data parameters (SPNs), transmission rates, and other related information are published in the digital annex (SAE J1939-DA).

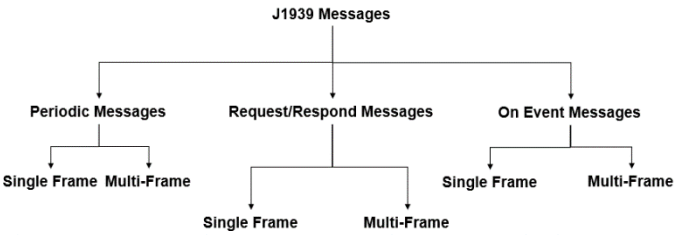

Figure 6: Taxonomy of J1939 messages used to span examples for implementation.

The SAE J1939 vehicle protocol can be divided into three categories based on occurrence characteristics, as shown in Figure 6. Periodic messages are repeatedly transmitted by the transmitting node at a cycle time interval. Request/Respond messages are messages that appear only upon request. The SAE J1939-DA specifies if a J1939 message is periodic, a request/response, or an on-event message. If the PGN is periodic then the cycle time is as well-defined here. The last category of messages that we evaluated was on event messages which are by default absent on an in-vehicle network and appear only upon a certain event occurrence. Each of these three categories can be further divided based on the message length. Messages that fit in 8 bytes of data are single frame as these only require one frame to complete the information exchange. Any length of the message that is greater than 8 bytes is called a multi-frame message.

Attacking J1939 Messages

The passthrough version of the shim DLL was updated to look for specific messages coming from the legitimate Vendor X DLL and passed to the application. These messages were manipulated to falsify what the ECU was reporting. The next couple of sections describe these message manipulations.

Periodic, Single Frame Message

Among the various periodic messages that we saw that was getting transmitted, we choose to tamper a single frame Vehicle Distance (VD) PGN 65248, and it reports critical information of accumulated distance travelled by the vehicle during its operation. As the ECU we had was a test ECU, the value reported by the ECU under normal operating condition was 0. In the developed malicious shim application, we wrote an algorithm to filter PGN 65248 and modify SPN 245 with a false value of 526,385,100 km. The sequence diagram reflecting the CAN trace log is shown in Figure 7. proves that our attack was successful and further the accumulated distance displayed on the diagnostic tool reflected the manipulated change.

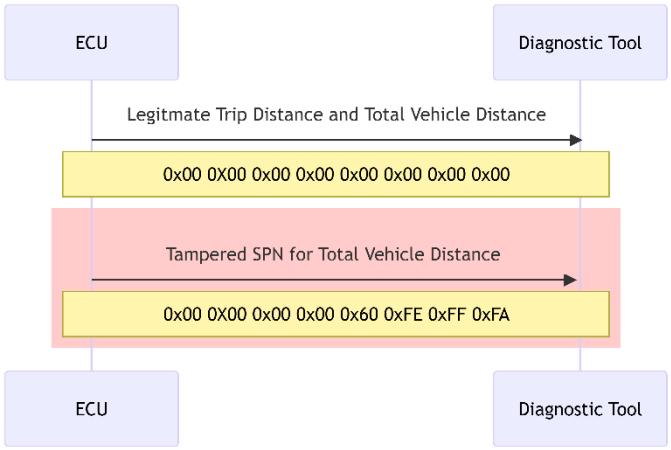

Figure 7: Sequence diagram that reflect log files showing the manipulation of SPN 245, total vehicle distance. The legitimate message has all zeros as the ECM used was brand new.

Periodic, Multi-Frame Messages

Per SAE J1939-73, the diagnostic condition of the controller application (CA) is conveyed to other nodes on the network via PGN 65226 – Diagnostic Message 1 (DM1) with diagnostic indicator lamp status and a series of SPNs and failure mode indicators (FMIs). A DM1 message is always transmitted, regardless of the presence or absence of diagnostic trouble codes (DTCs), every second. The test ECU shown in Figure 5 indicated there were 5 DTCs that had the following decimal values: 17039451, 17039460, 16973934, 17039771 and 16974996. Because there was more than 1 DTC, the DM1 message was packed into a multi-frame message using the SAE J1939 Transport Protocol.

The malicious shim application was able to modify the DTCs to 3450536027, 4024303716, 3148480622, 3721134491 and 4278256641, as shown in Figure 8. Thus, the diagnostic condition of the ECU is falsified and displayed in the diagnostic tool.

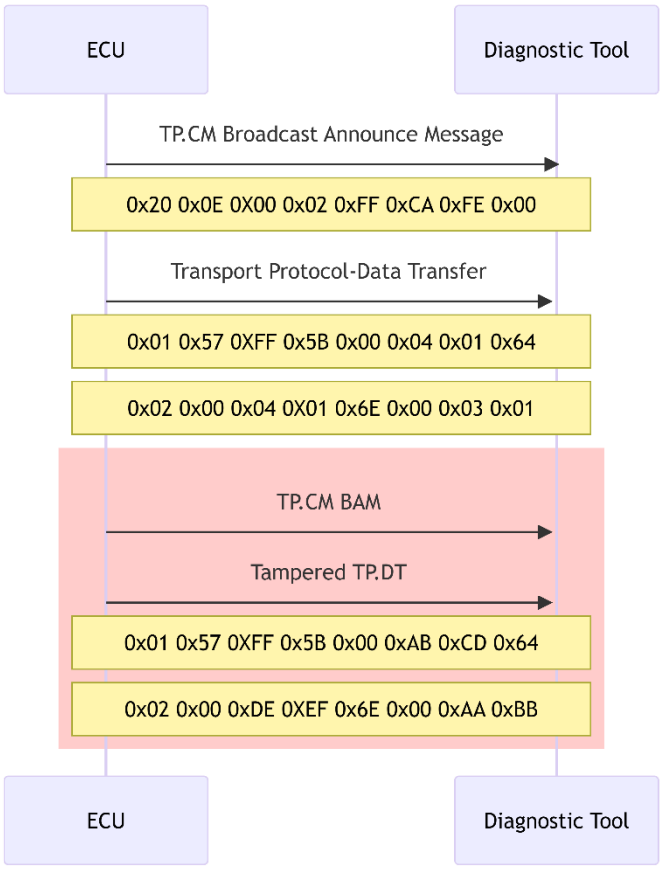

Figure 8: Demonstration of manipulating multi-frame messages in J1939 with the DM1 message as an example.

We can henceforth conclude that both single frame and multi-frame periodic messages are vulnerable to RP1210 based MitM attacks with the shim DLL.

Request/Respond Messages

PGN 65253, engine hours and revolutions were chosen for tampering. This is a single frame, on-request message. The accumulated time of operation of engine and accumulated number of revolutions of engine crankshaft during the operation are reported in this PGN via SPN 247 and 249. As we were using a new test bench ECU, SPN 249 reported a value of 0. Using our malicious shim application, we detected when the response PGN was transmitted by the ECU. Post detection, we modified with false value of 4211081000 revolutions, as shown in Figure 9.

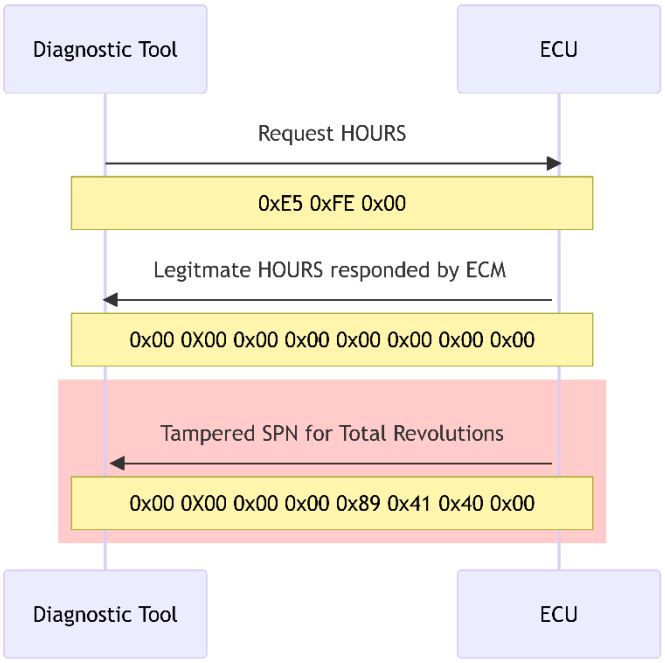

Figure 9: Example of manipulating an on-request message for engine revolutions. The legitimate message has all zeros as the ECM used was brand new.

Vehicle identification number is a 17-byte character string assigned by a manufacturer and transmitted in J1939 in ASCII which has length greater than a single frame can accommodate. In the test ECU, we had the vehicle identification number set to the printed number zero in ASCII, 0x30 with a length of 17 bytes. Using the malicious shim application, the vehicle identification number was changed to an ASCII text that read “HACKEDBYSHARIKA|0,” as shown in Figure 10. Hence, we conclude our attack evaluation of request/respond messages that the MitM affects both single and multi-framed request/respond messages.

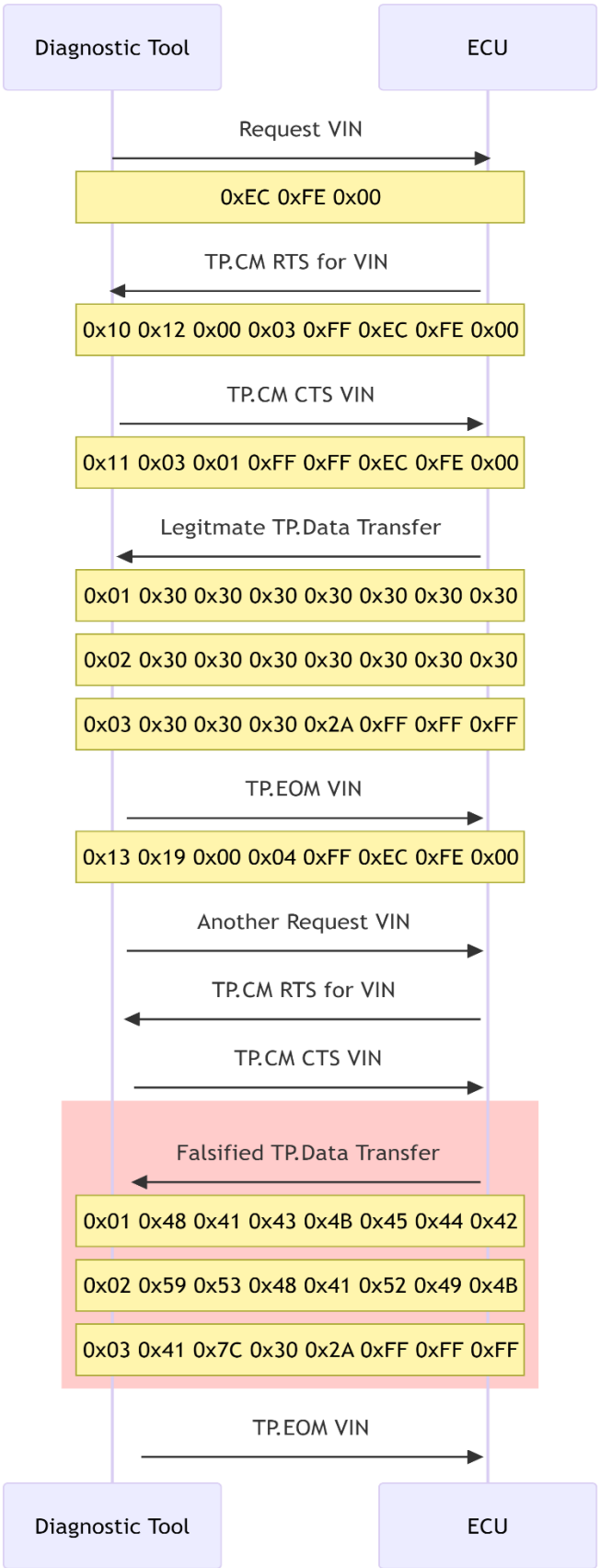

Figure 10: Sequence diagram that reflect log files manipulating a VIN, which is a requested multi-frame message.

On-Event Messages

Per SAE J1939, every controller application (CA) is mandated to claim their source address upon powerup and if there is any change to CA's source address or name. This message can be considered as an on-event message as a transmitting node transmits this out when it goes through the power up initialization event. The ECU that we had in our experimental set-up had a manufacturer code of 0x55. We detected an address claim response message from the Engine Control Module (ECM) and replaced the manufacturer code as 0x15, as shown in Figure 11. This stealthy manipulation may make the software show a different manufacturer of the ECU than it really is.

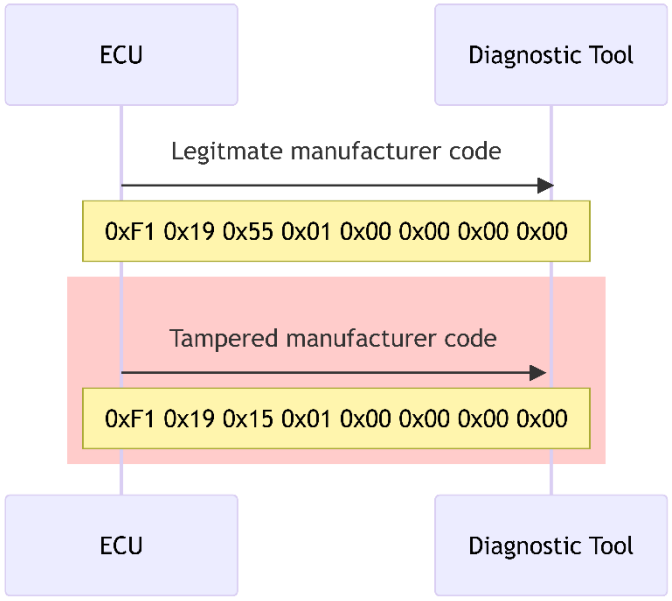

Figure 11. Sequence diagram that reflects log files to change the data in the Address Claim in the NAME field defined in SAE J1939-81.

For brevity, event driven multi-frame messages are omitted. However, these messages are common on events driven by diagnostic applications, since a user requesting a parameter or uploading firmware would be event driven. Even with the absence of this example, sufficient evidence is presented to realize that any type of J1939 message flowing through the RP1210 stack can be manipulated. This, of course, means an attacker has many options to exercise a nefarious plot.

Cyber Defense for Diagnostic Interfaces

An ECU software product lifecycle starts at requirement and architectural design, followed by implementation and unit testing. Various subcomponents are integrated, and a system validation test is performed to release a software product. The cost of fixing software related vulnerabilities found during initial stages of the software product lifecycle is less compared to the vulnerabilities found during later stages of a product lifecycle [16]. ISO/SAE 21434 recommends cybersecurity engineering by design achieved through establishing cybersecurity policies, culture, management, confirmation, supporting processes, and ensuring cybersecurity lifecycle development and processes. Following these guidelines, we did not produce a single patch solution to the attacks that we discovered, instead we produced a more concrete mitigation mechanism that can protect all J1939 messages.

The latest ISO 14229 [22] specification released has considered cybersecurity by design aspects and has enhanced built-in security features into the Unified Diagnostic Service (UDS) protocol. UDS supports secure transmission service over service identifier (SID) 84$. As classic CAN is limited to 8 bytes, it is often inadequate to support encrypted communication as encrypting messages often demands a higher data throughput. Emerging CAN-Flexible Data [23] (CAN-FD) with security trailer is the most appropriate mitigation to the attacks demonstrated. SAE J1939-22 defines safety/security trailer to be appended to PGN data that is fit into a container parameter group (C-PG). With a Message Authentication Code (MAC) appended on the transmitted J1939 messages a receiving node can ensure integrity of the data that is received. As a pre-shared secret key is associated with a MAC it would not be possible for the MitM attacker in the shim to match the MAC and make the manipulated frame look legitimate. Additionally, SAE J1939-22 offers multi-PGN packing which further improves the data throughput efficiency. We anticipate that secure on-board communications with optional encryption would get formalized in CAN FD Network Security standard which is currently work in progress [24] for SAE J1939 networks.

Given the data throughput limitation of classic CAN and SAE J1939 unencrypted communication, we propose the below mitigation mechanism. As the security aspect that is compromised is integrity and confidentiality we geared towards an authenticated encryption-based solution. The basic idea, shown in Figure 12, is in addition to the legitimate message, and ECU will transmit a security validation message that the receiver can use to verify if the legitimate message is tampered with or not. If the verification fails, the receiver shall simply discard the received frame. In our model's simplest form, the security message could contain a MAC that is generated by the freshest or latest message transmitted out. The security message is expected to alter regarding data changes in legitimate message. With the addition of security message there is an impact on data throughput, but this is trade-off that our model and several existing models accept.

Figure 12. Mitigating undetected message manipulation.

In our demonstration, we performed encryption and authentication using AES-128 in GCM mode. However, the major challenge of securing SAE J1939 messages is maintaining the standard's compatibility. Given the diversified nature of security use cases and considering the goal of holistically applying security, we elaborate the security message by adding extra header information indicating security properties for flexibility of use. Inspired by ISO 14229 secured data transmission, we added a header within data security parameter along with the message, cipher, or the authentication tag. On exploring SAE J1939 specifications, we found that the data security parameter group PGN 54272, Diagnostic Message 18 (DM18) fits our use case. Per SAE J1939-73, DM18 is used to send security entities of a given type and length where entities are data procedures to ensure data security. The DM18 message is detailed in Table 2. SAE J1939-73 defines security entity type of values 0 – 3 and rest as reserved. We used this reserved space to indicate the message type as encrypted or signed and if a pre-shared or shared key is used.

Within the data security parameter, we defined a byte for algorithm identification. In our demonstration we used a value of 0x13 to indicate AES-128 in GCM mode. The field of signature length is used if we are signing the message if not can be left at 0. To prevent the message from replay attacks we added a replay counter; however, for our demonstration we are not using this field. The message/cipher field shall contain the message in plain text if signed or the cipher if encrypted.

A sequence diagram and a network trace showing an example of using DM18 as an additional security message for vehicle identification message PGN 65260 are shown in Figure 14 and Figure 15 in the Appendix. In this example, we are encrypting and adding a tag and hence we used a value of 0x0B for security entity type. Even though transmitting a tag would mitigate the attack sufficiently at lowest overhead, we have given options to encrypt, and to transmit freshness value as other mitigation options.

Table 2. Data Security Message (DM18) updates for defense.

Byte Pos.	Bits	Definition (Existing in the SAE J1939-73)	Updates to existing definition
1	8-1(LSB)	Security Entity Length – Length of the data security parameter	
2	8-5(MSB)		
2	4-1	Security Entity Type – Indicating type of usage 0000 – Data is long seed 0001 – Data is long key 0010 – Data is a session key 0011 – Data is a certificate 0100 – 1111 - Reserved	1000 – Data is encrypted with pre-shared key 1001 – Data is signed with pre-shared key 1011 – Data is encrypted and signed with pre-shared key 1100 – Data is encrypted with dynamically derived key 1101 – Data is signed with dynamically derived key 1111 – Data is encrypted and signed with dynamically derived key
3	8-1	Data Security Parameter	Signature/Encryption Calculation – Contains an algorithm identifier
4-5	8-1		Signature Length – Length of signature portion of the message
6-7	8-1		Anti-replay Counter – Incrementing counter to prevent replay attack
8- n*	8-1		Message/Cipher
n+1 – m** n+ Signature Length	8-1		Signature
n * = Message/Cipher Length m** = n+ Signature Length			

To holistically apply security to all J1939 messages we built a SAE J1939 security sublayer on both controller side and diagnostics side. On the ECU side, the SAE J1939 sublayer would take the raw data

that the application wants to transmit out and apply the needed security and transmit secure message out. The transmitter shall indicate the cryptographic operations performed to meet the security requirement in the header. While on the receiver side the SAE J1939 security sublayer would look up the header and apply the needed cryptographic actions to extract the plain text and pass it on to the diagnostic application. When security is implemented as a security sublayer, there is no change that needs to be made on the controller application or diagnostic application and hence is the most efficient way to do security. This approach eliminates the need to change any designs for RP1210 as it treats both the J1939 network and the RP1210 stack as untrusted entities, as shown in Figure 13.

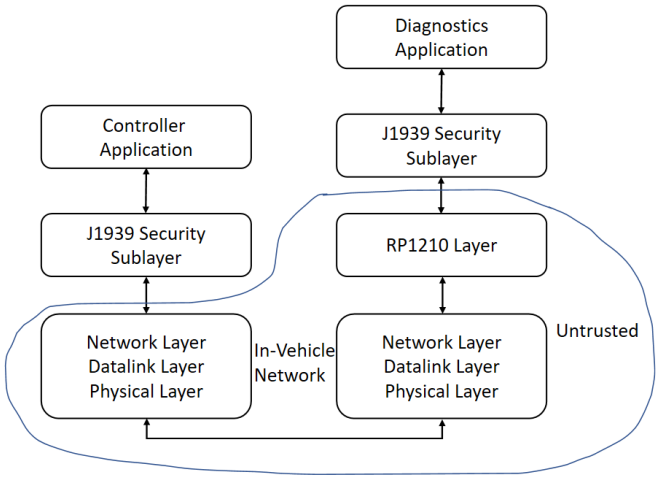

Figure 13. Proposed security architecture where external layers are untrusted.

The J1939 Security Sublayers on each side of the communication stack are controlled by the same vendor. The diagnostics application is written for the specific controller application from the same company. This means there is an ability to pre-share keys or manage security controls without the need for cooperation from third parties.

Conclusion

Based on automotive cybersecurity statistics, there is a dramatic increase in the number of automotive cybersecurity incidents. Among the attack vectors used, about 8% were through the diagnostic port where diagnostic tools are connected. Many efforts have been made on attacking CAN and SAE J1939, but most needed access to the CAN bus, so it was difficult to make the attack stealthy if the attacker is an outsider. However, if a computer used by a trusted maintenance technician is use for the attack, physical access to the CAN bus achieved with the trusted maintenance actions. This work demonstrated how to insert a shim DLL to launch Machine-in-the-Middle (MitM) attacks on communications between the service tool and diagnostics software. Since the attacks can read and write in both directions, the attacker, through the MitM has remote access to the CAN bus. It was shown that MitM attacks are possible on different types of SAE J1939 messages. Also, we demonstrated the importance of holistic mitigation approaches by using a security sublayer architected in the design phase of a software lifecycle.

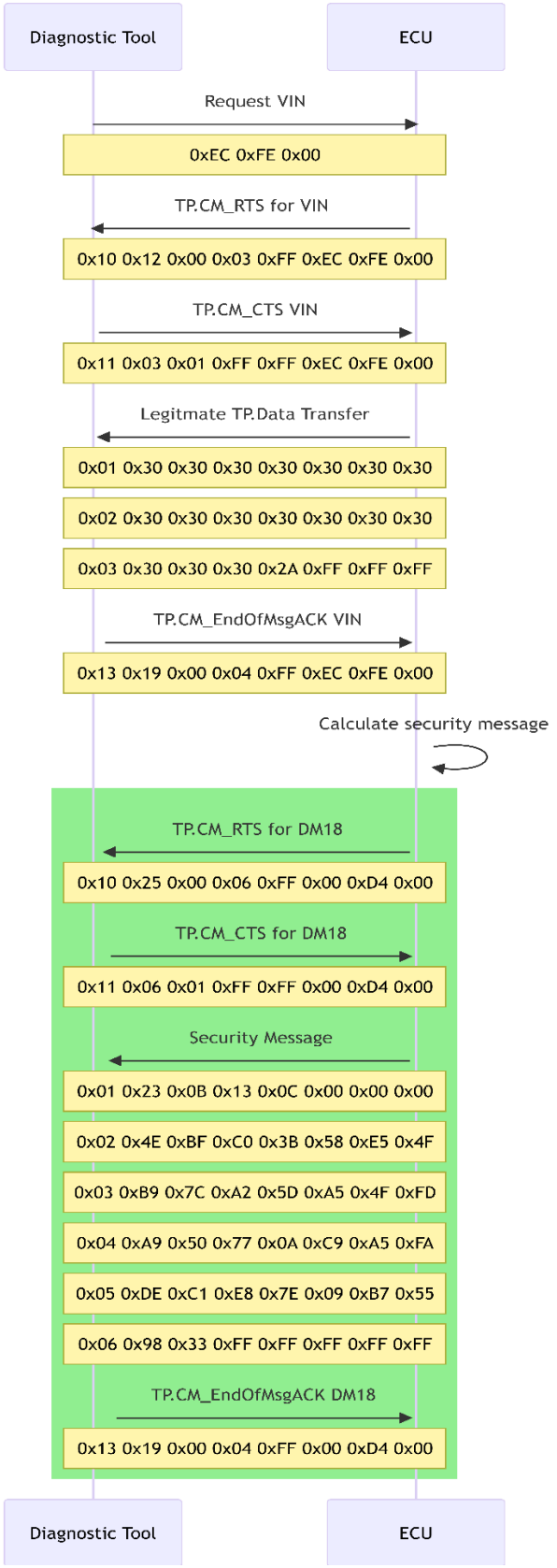

Figure 14: Sequence diagram that reflect log files showing the utility of DM18 to send secure messages over SAE J1939.

References

1. Burakova, Y. and Hass, B., Millar, L. and Weimerskirch, A., “Truck Hacking: An Experimental Analysis of the SAE J1939 Standard”, 10th USENIX Workshop on Offensive Technologies 2016.
2. Becker, S., “A Heavy Vehicle Network Security Evaluation”, Master’s Thesis. December 2, 2016, University of Michigan.
3. Hoppe, T. and Dittman, J., “Sniffing/replay attacks on CAN buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy,” In Workshop on Embedded Systems Security (WESS), 2007.
4. Merco R., Biron, Z. A. and Pisu, P., "Replay Attack Detection in a Platoon of Connected Vehicles with Cooperative Adaptive Cruise Control," 2018 Annual American Control Conference (ACC), 2018, pp. 5582-5587, https://doi.org/10.23919/ACC.2018.8431538
5. Matsubayashi M., et al., "Attacks Against UDS on DoIP by Exploiting Diagnostic Communications and Their Countermeasures," IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), pp. 1-6, 2021, https://doi.org/10.1109/VTC2021-Spring51267.2021.9448963
6. Koubâa, A., et al., “Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey,” IEEE Access 2019; 7:87658–80 https://doi.org/10.1109/ACCESS.2019.2924410
7. Piètre-Cambacédès, L. and Bouissou, M. “Cross-fertilization between safety and security engineering,” Reliability Engineering & System Safety, vol. 110, p. 110–126, 02 2013, https://doi.org/10.1016/j.ress.2012.09.011
8. Mukherjee, S., Shirazi, H., Ray, I., Daily, J. and Gamble, R., “Practical DoS attacks on embedded networks in commercial vehicles,” 2016 International Conference on Information Systems Security (ICISS 2016), pp. 23–42, 2016, https://link.springer.com/content/pdf/10.1007/978-3-319-49806-5.pdf
9. Gazdag, A., Ferenczi C., Buttyán L., “Development of a Man-in-the-Middle Attack Device for the CAN Bus”, Proceedings of the 1st Conference on Information Technology and Data Science Debrecen, Hungary, November 6–8, 2020, http://www.hit.bme.hu/~buttyan/publications/GazdagFB2020citds.pdf
10. Daily, J., Nnaji, D. and Ettlinger, B., “Demo: Securing Heavy Vehicle Diagnostics”, Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021 25 February 2021, https://doi.org/10.14722/autosec.2021.23020
11. Buttyán, L., “Hacking cars in the style of Stuxnet – CrySyS Blog.” https://blog.crysys.hu/2015/10/hacking-cars-in-the-style-of-stuxnet/ last accessed 14 Dec. 2022.
12. Daily, J. and Kulkarni, P., “Secure Heavy Vehicle Diagnostics”, In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2020.
13. Murvay, P. -S. and Groza, B., "Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol," in IEEE Transactions on Vehicular Technology, vol. 67, no. 5, pp. 4325-4339, May 2018, https://doi.org/10.1109/TVT.2018.2795384
14. Rogers, M., Weigand, P., Happa, J. and Rasmussen, K., "Detecting CAN Attacks on J1939 and NMEA 2000 Networks," in IEEE Transactions on Dependable and Secure Computing, 2022, https://doi.org/10.1109/TDSC.2022.3182481
15. Dadam, S. R., Zhu, D., Kumar, V., Ravi, V., & Palukuru, V. S. S., “Onboard Cybersecurity Diagnostic System for Connected Vehicles,” SAE Technical Paper Series, 2021-01-1249, https://doi.org/10.4271/2021-01-1249
16. Campo, M., Mukherjee, S., and Daily, J., "Real-Time Network Defense of SAE J1939 Address Claim Attacks," SAE Int. J. Commer. Veh. 14(3):319-328, 2021.
17. Thompson, M., "UDS Security Access for Constrained ECUs," SAE Technical Paper 2022-01-0132, 2022, https://doi.org/10.4271/2022-01-0132
18. ISO/SAE 21434: Road Vehicles – Cybersecurity Engineering”, ISO/SAE International Standard, First edition 2021-08 https://www.sae.org/standards/content/iso/sae21434/
19. RP1210D, Technology and Maintenance Council Engineering Recommended Practice, Revision 4.06, 2022.
20. “Introduction to RP1210A (RP1210B),” Kvaser, https://www.kvaser.com/about-can/can-standards/rp1210/, last accessed 27 Jan 2023.
21. Gardiner, B., “Commercial Transportation: Truck Hacking Slides”, NMFTA CTSRP Documents, pp.1–90, September 2021, https://biz.nmfta.org/documents/ctsrp/Actionable_Mitigations_Options_v9_DIST.pdf, last accessed 27 Jan 2023
22. Technical Committee ISO/TC 22/SC 31, “ISO 14229-1: 20220, Road Vehicles – Unified Diagnostic Services (UDS) - Part 1: Application Layer”, ISO standard edition-3, 2020-02, https://www.iso.org/standard/72439.html
23. SAE J1939-22: CAN FD Data Link Layer, SAE International, https://doi.org/10.4271/J1939-22_202209, 2022
24. J1939-91C: CAN FD Network Security, SAE International, unpublished work in progress, 2022
25. J1939-71: Vehicle Application Layer, SAE International, https://doi.org/10.4271/J1939/71_202208, 2022
26. “Global Automotive Cybersecurity Report 2023,” *Upstream Security*, 11 Aug. 2022, https://upstream.auto/resources/. last accessed 27 Jan 2023.

Contact Information

Sharika Kumar, sharkia.kumar@cummins.com
Jeremy Daily, jeremy.daily@colostate.edu

Acknowledgments

The authors would like to acknowledge DG Technologies for supplying sample source code for RP1210 and sharing their expertise, especially Angela Adelsberger.

Definitions/Abbreviations

AD	Analog to Digital
ATA	American Trucking Association
CAN	Controller Area Network
CAN-FD	Flexible Data Rate CAN
CRC	Cyclic Redundancy Check
CTS	Clear to Send
DA	Destination Address
DLL	Dynamic Link Library

DOS	Denial of Service	**PDU**	Protocol Data Unit
ECDH	Elliptic Curve Diffie-Hellman	**RP**	Recommended Practice
ECM	Electronic Control Module	**RTS**	Request to Send
ECU	Electronic Control Unit	**SA**	Source Address
ISO	International Organization for Standardization	**TLS**	Transport Layer Security
IT	Information Technology	**TMC**	Technology Maintenance Council
MHD	Medium and Heavy-Duty	**TRNG**	True Random Number Generator
MitM	Machine-in-the-Middle	**UDS**	Unified Diagnostics Services
OBD	Off-board Diagnostics	**VDA**	Vehicle Diagnostics Adapter
PC	Personal Computer	**VIN**	Vehicle Identification Number
PGN	parameter group number	**VPN**	Virtual Private Network

Appendix

The following code snippet demonstrates how to compose C code to modify an RP1210_ReadMessage command from the shim DLL. This example reads the buffer using the known legitimate DLL and manipulates the buffer after it has been read. The manipulation only takes place on the PGN of 0x00FEE1, which is vehicle distance. The status return is the same as it was from the legitimate API.

```
short FUNCTION_MODIFIER RP1210_ReadMessage( short nClientID,
                                            char far  *buf,
                                            short nBufferSize,
                                            short nBlockOnRead)
{
    int writeSize;
    if(xx_DLL.functions.readMessage){
        status = xx_DLL.functions.readMessage( nClientID,
                                               buf,
                                               nBufferSize,
                                               nBlockOnRead);
    }
    else {
         status = -1;
    }
    if (status > 0){
      size_t i = 0
      //Total Vehicle Distance -- byte manipulations
      if ((buf[4] == 0xE1) && (buf[5] == 0xFE) && (buf[6] == 0x00)) {
        buf[9] = 0xCC;
        buf[10] = 0xBB;
        buf[11] = 0xAA;
    }
    return(status);
}
```

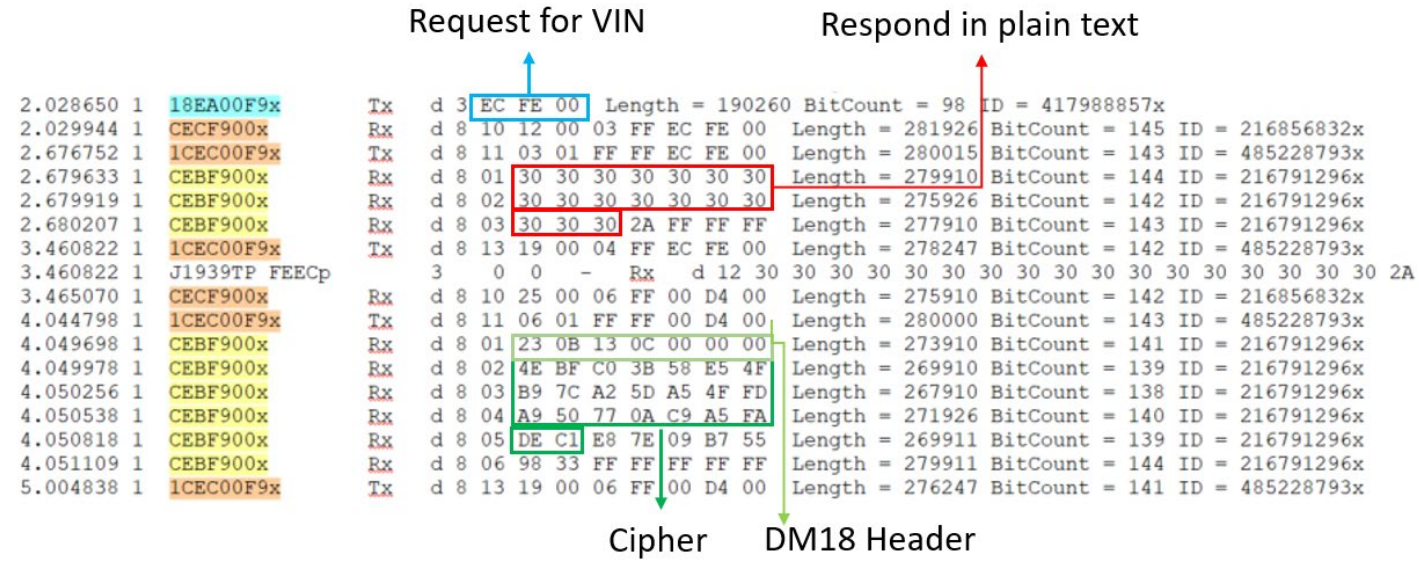

Figure 15: Network trace showing the utility of DM18 to send secure messages over SAE J1939.